Artificial Intelligence 173 (2009) 197–220

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Automatic interpretation of loosely encoded input ☆

James Fan [a,*], Ken Barker [b], Bruce Porter [b]

[a] IBM TJ Watson Research Center, 19 Skyline Dr., Hawthorne, NY 10532, USA
[b] Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712, USA

ARTICLE INFO

Article history:
Received 14 November 2007
Received in revised form 12 October 2008
Accepted 20 October 2008
Available online 29 October 2008

Keywords:
Knowledge based systems
Question answering
Metonymy
Noun compound

ABSTRACT

Knowledge-based systems are often brittle when given unanticipated input, i.e. assertions or queries that misalign with the ontology of the knowledge base. We call such misalignments "loose speak". We found that loose speak occurs frequently in interactions with knowledge-based systems, but with such regularity that it often can be interpreted and corrected algorithmically. We also found that the common types of loose speak, such as metonymy and noun-noun compounds, have a common root cause. We created a Loose-Speak Interpreter and evaluated it with a variety of empirical studies in different domains and tasks. We found that a single, parsimonious algorithm successfully interpreted numerous manifestations of loose speak with an average precision of 98% and an average recall of 90%.

1. Brittleness in interaction with knowledge-based systems

Many AI tasks – such as natural language understanding and expert reasoning – require formal encodings of substantial bodies of knowledge, yet this requirement can cause AI systems to be brittle, sometimes to the point that the systems work for only limited, well scripted examples. Many factors contribute to the brittleness of knowledge-based systems – such as gaps in the knowledge base or reasoning methods that are unsound or incomplete – so there is no easy solution. Nevertheless, in this paper we identify a frequent source of brittleness, show the ways it can arise and demonstrate that a single solution can fix all of the ways it appears.

Here we are concerned with the interface between the knowledge base and its user, which might be a person or an application program. This is the first point at which failures might occur, and, if they are not resolved, they will cause further problems downstream.

As background, the user interacts with a knowledge base through *ask* and *tell* primitives. The user expresses the query or assertion by referencing concepts and relations. As a simple example, consider this query which might be asked of a hypothetical chemistry knowledge base:

```
(ask
  '(the result of
    (a Chemical-Reaction with
      (reactants (Water and Hydrogen)))))
```

☆ Support for this research is provided by contracts from SRI International and Vulcan Inc.

* Corresponding author.
E-mail addresses: fanj@us.ibm.com (J. Fan), kbarker@cs.utexas.edu (K. Barker), porter@cs.utexas.edu (B. Porter).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2008.10.007

Forming queries is not easy because they must align with the ontology of the knowledge base. The ontology consists of terms, such as *Chemical-Reaction* and *Water*, as well as relations, such as *result* and *reactants*. Queries will fail if they misalign with the ontology – even in seemingly minor ways. For example, the query might refer to H_2O instead of *Water*, or (more pernicious) the terms and relations might match lexically, but differ semantically. The same issues arise with *tell* operations.

In practice, it is difficult for users to align queries and assertions with the ontology of the knowledge base because it requires them to know the internal structure of the knowledge base (and to exercise great care). This is problematic for two reasons. First, principles of modularity suggest that users should be able to treat a knowledge base as a "black box" utility, even to the degree that one knowledge base can be plugged in for another. Second, knowledge bases, like many engineered systems, are complex and full of idiosyncratic design decisions. Even the knowledge engineers who designed the ontology can have trouble using it.

This became apparent in the Pilot Phase of Project Halo [66], in which a team of knowledge engineers built a knowledge base to answer Advanced Placement chemistry questions. After the knowledge base was built, during the evaluation phase, the same team encoded about 150 questions by translating them from English (as they appear on the exam) into the formal language and ontology used by the knowledge base. This task required two weeks, in large part because of the difficulty of aligning the formal encodings of the exam questions with the knowledge base. Without proper alignment, few if any of the question encodings would be answered correctly [22].

For example, consider the following simple question from Project Halo:

Which of the following compounds is insoluble in water?
(a) $Pb(NO_3)_2$
(b) Li_2CO_3
(c) $(NH_4)_3PO_4$
(d) $Ba(OH)_2$
(e) $BaSO_4$

The problem with this question is its underspecification. Do the five options (a)–(e) refer to individual molecules or to chemicals comprised of these molecules? A chemical (which is a set of molecules) has properties, such as quantity, physical state and solubility, that a single molecule does not have. However, in chemistry a chemical formula can be used to denote a single molecule in some scenarios and a set of molecules in others. If the query is incorrectly encoded as asking about the solubility of a single molecule, then no correct answer will be found.

Much of the prior research on aligning queries and assertions with the knowledge base's ontology is motivated by the challenges in a neighboring area of research: natural language understanding. Because natural languages are so expressive, they allow the same content to be expressed in many ways. Furthermore, expressions are commonly underspecified, which requires the recipient to infer missing information. Common examples include:

- metonymy (reference to an entity or event by its components; See Section 3.2.1).
- noun compounds (a sequence of nouns without explicit semantic relations between them; See Section 3.2.6).

There has been considerable research on computational methods for natural language understanding, but most of the research has treated these issues separately from one another. In contrast, we consider them to have a common core: in each case the recipient of a natural-language expression faces the challenge of aligning it with his own ontology. We propose a computational solution for this common problem and we show through a series of experiments that it performs well for a variety of hard problems in language interpretation.

1.1. Paper outline

Before we discuss our solution in detail, we need to define our terms. Section 2 defines "loose speak" as an input to a knowledge-based system (*ask* and *tell* operations) that misaligns with the ontology of the knowledge base. Section 2 also states our thesis: although loose speak takes many forms, it can be resolved with a single, algorithmic method.

Our optimism stems in part from our study of the many different occurrences of loose speak in natural texts and our estimation of their frequencies. Section 3 presents the results of our study and describes the most common types of loose speak in detail.

We found that common types of loose speak follow a regular pattern and can be handled with a uniform solution, which we have implemented in the "Loose-Speak Interpreter". Section 4 describes the interpreter's input, output, algorithm, and key components.

We evaluated the interpreter on various tasks and data sets in different domains. Section 5 describes three evaluations. Together they show that the Loose-Speak Interpreter is highly effective and general, in the sense that it does not require a large body of domain-specific knowledge.

Section 6 surveys related work. Compared with work in related areas, the Loose-Speak Interpreter is a unified solution to a variety of problems, such as metonymy resolution and noun compound interpretation.

Section 7 summarizes the paper and outlines future work.

2. Our approach

Before we propose a solution to the loose-speak interpretation problem, let us define some useful terms.

2.1. Terminology

During a knowledge base interaction session, an agent presents assertions or queries to the knowledge-based system by encoding them in the interaction language of the system. If the assertion or query is encoded without regard for the knowledge-based system's ontology, then we call it a *naïve encoding* of the input. If, on the other hand, the encoding not only conveys the meaning of the input, but is also compatible with the ontology, then we call it a *correct encoding* of the input.

Loose speak is defined as the communication of the naïve encoding of a question or assertion when that encoding differs from its correct encoding. We call the task of resolving this difference – i.e., the task of aligning a knowledge encoding with the ontology of a knowledge base – *loose-speak interpretation*. The goal of our project is to improve knowledge base interaction by developing ways to automate loose-speak interpretation.

This is not to say that naïve encodings are all bad. In fact, they are typically the most straightforward and most faithful encoding of a piece of knowledge. Fidelity is a basic requirement when interacting with knowledge-based systems because without it the *ask* and *tell* operations may include unintended semantics. Fidelity was so important in the Pilot Phase of Project Halo [66], for example, that it was separately evaluated by a panel of knowledge representation and reasoning researchers.

This example illustrates both the fidelity of naïve encodings and the problems that they cause:

(1) A water solution of *HCl* is highly conductive.

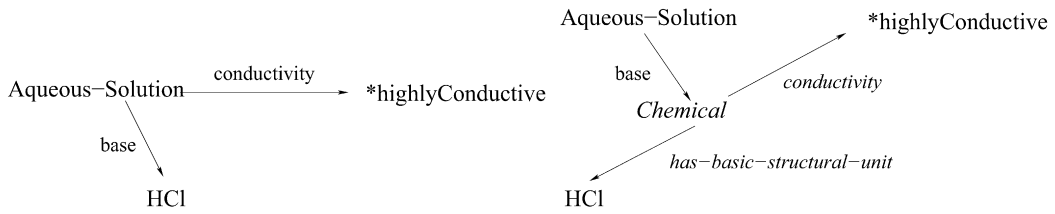
A naïve encoding may look like Fig. 1(a) in conceptual graph form. Although it appears to be faithful to the original input, whether or not it is loose speak depends upon the knowledge base to which it is asserted. For example, when it is asserted to the chemistry knowledge base we built for Project Halo, the encoding is indeed incorrect, as it has two problems. First, it uses an individual to stand for an aggregate: a single molecule, such as *HCl*, cannot act as the solute of a solution because it does not have properties such as quantity. The chemistry knowledge base uses the *has-basic-structural-unit* relation to relate an aggregate to the type of individual of which it is comprised. The *HCl* in the encoding should be replaced by a *Chemical* whose basic structural unit is a *HCl* molecule. Second, it uses metonymy: the *conductivity* property belongs to solutes (not solutions) because it is the solutes that determine the conductivity of a solution. The *conductivity* property should be placed on the base of the Aqueous-Solution instead. Fig. 1(b) shows the correct encoding of the example. (Section 3.2 describes these types of loose speak in detail.)

As the example illustrates, naïve encodings may be the most faithful encodings of input knowledge, but they may not be useful for automated reasoning because they do not align properly with the knowledge base on which the reasoning methods depend.

2.2. Thesis

The thesis of our work is that: misalignments between naïve encodings and correct encodings occur frequently during both *ask* and *tell* operations. However, these occurrences follow regular patterns and they can be resolved with a single algorithmic method.

To evaluate this thesis, we analyzed five corpora – three containing assertions and two containing questions (see Section 3.1 for details). The purpose of the study was to measure the frequency of occurrences of loose speak and to categorize them. We found loose speak to be very common – occurring in 72–100% of sentences. We found six types of loose speak, including ones similar to long studied natural language phenomena, such as metonymy and noun compounds.



(a) The naïve encoding. (b) The correct encoding.

Fig. 1. The naïve and correct encoding of "a water solution of *HCl* is highly conductive".

We set the goal of identifying the commonality among all these types of loose speak and devising a single method for resolving them. Furthermore, we set the goal that the method would require using only the knowledge from the knowledge base involved in the interaction; no special "loose-speak interpretation knowledge" would be required. If loose-speak interpretation required additional knowledge, such as detailed knowledge about the idiosyncrasies in the knowledge base, then any gains in knowledge base interaction by loose-speak interpretation systems would be erased by the efforts needed to create the additional knowledge, and loose-speak interpretation would be of little value in reducing the brittleness of knowledge-based systems. Additionally, the computational method should not be bound to a specific domain or a knowledge base.

We believed these goals could be achieved because our corpora analysis revealed that naïve encodings are very similar to their correct encodings, differing in relatively few ways. Consequently, the types of loose speak we discovered could be resolved by a computational method that searches the space of encodings near the naïve one; the correct encoding would be found within that space.

3. Types of loose speak

The previous section explained how loose speak originates from the idiosyncrasies of an engineered knowledge base, which raises the concern that occurrences of loose speak are boundless. In this section, we present a study that categorizes loose speak (Section 3.1). The study shows that although loose speak is very common, it concentrates on a small number of types. We describe the most frequently used types of loose speak in detail (Section 3.2).

3.1. Frequency study

We evaluated the frequency of loose speak occurrences and categorized them through a corpus study. First, we investigated loose speak frequencies in assertions (*tell* operation) based on three corpora of texts written in English. Natural language texts contain a large amount of assertional knowledge, and they are often used as knowledge sources during the process of knowledge acquisition.

The three corpora we used were the Brown corpus, the MUC corpus and the Alberts corpus. The Brown corpus [37] consists of one million words of American English texts printed in 1961. The texts for the corpus were randomly sampled from 15 different domains. The Alberts corpus is a college-level cell biology text [1] consisting of 0.45 million words. The MUC 3 and 4 corpus is the data set used in the third and fourth Message Understanding Conference [15,16]. It consists of 0.56 million words from news stories.

As stated in Section 2, loose speak occurs in the context of an existing knowledge base ontology, hence the frequencies of different types of loose speak depend on the ontology used. For our study, we used the domain-independent ontology of the Component Library [6]. This ontology provides the typical top-level distinctions between entities, events, and roles, and a relatively small set of relations among these types. Although studies based on other knowledge base ontologies may yield different frequencies, the results of our studies give a qualitative estimate of the distribution of frequencies of different types of loose speak.

Second, we investigated loose speak frequencies in queries based on two corpora from Project Halo [66]. Two sets of Advanced Placement test questions (*ask* operation) were used during the project, and both of them contain many queries. Again, we used the Component Library as the ontology to determine the loose speak in both sets of questions, and we compared naïve encodings of the queries (i.e. encodings done by users without knowledge of the Component Library) with the correct encodings (i.e. encodings done by users with knowledge of the Component Library).

Table 1 reports the results of our study. We can draw several conclusions from this study. First, loose speak is ubiquitous. The incidence of loose speak varies from 72% (MUC corpus) to 100% (Halo Final questions). Second, loose speak does concentrate on a small number of types. We found six common types of loose speak, some of which occur as frequently as 87% of the time. Third, different domains have different distributions of types of loose speak, as illustrated by the significant differences across our corpora.

Table 1
The frequencies of loose speak occurrences in the three corpus samples and in the questions from Project Halo. The sum of the loose speak type frequencies exceeds 100% because each sentence in both the corpus studies and the Halo study may contain more than one loose speak type.

	Brown	Alberts	MUC	Halo Pilot	Halo Final
Sample size	99	96	100	44	102
No loose speak	20.2%	11.5%	28.0%	2.3%	0.0%
Metonymy	7.1%	1.0%	18.0%	25.0%	10.8%
Causal factor	0%	0%	0%	45.5%	23.5%
Aggregate	5.1%	4.2%	2.0%	77.3%	87.3%
Role	14.1%	13.5%	27.0%	9.1%	7.8%
Overly generic	0%	0%	0%	34.1%	40.2%
Noun compounds	50.5%	68.8%	86.0%	22.7%	29.4%
Other types	11.1%	5.2%	1.0%	4.5%	2.0%

3.2. Frequent types of loose speak

In the following sections, we describe the most frequently used types of loose speak.

3.2.1. Metonymy

Metonymy occurs when an entity or event (the referent) is referred to by a closely related concept (the metonym). Several researchers have identified "metonymic relations" which are commonly used to relate the referent with the metonym [2, 54]. Our work focuses on Lakoff and Johnson's list of relations [38], containing: PART-FOR-WHOLE, PRODUCER-FOR-PRODUCT, OBJECT-FOR-USER, CONTROLLER-FOR-CONTROLLED, INSTITUTION-FOR-PEOPLE-RESPONSIBLE, PLACE-FOR-INSTITUTION, and PLACE-FOR-EVENT.

As it is usually studied, metonymy occurs in natural language expressions. However, in our work on interactions with knowledge-based systems, we focused on metonymy in expressions in formal representations. While these phenomena share much in common, to avoid confusion, we will refer to the former as "linguistic metonymy", and the latter as simply "metonymy".

Fig. 1(a) contains an example of metonymy: a property (*conductivity*), which pertains to a part (solutes), is ascribed to the whole (the aqueous solution). Unlike the usual PART-FOR-WHOLE type of metonymy, this is a case of WHOLE-FOR-PART type of metonymy, which we found to be common in loose speak. An agent interacting with a knowledge-based system may use this type of metonymy when the part-whole hierarchy of the knowledge base differs from the agent's expectation of it. Part-whole hierarchies are a natural way to organize entities for reasoning, and different part-whole hierarchies are possible for the same set of concepts depending on the level of granularity. For example, a *Cylinder* can be considered as a part of an *Engine Block*, an *Engine* or an *Automobile*. Knowledge engineers typically choose one hierarchy which may not be the same as what a user has in mind.

Resolving this type of metonymy could be a heavy burden on a naïve user especially when a large number of parts is involved. For example, a knowledge base of airplanes implements a hierarchy of thousands if not millions of parts of an airplane, and browsing through the parts to ensure that the correct part-whole relation is used is no trivial task.

3.2.2. Causal factor

A causal factor type of loose speak occurs when one refers to an effect by one of its causes. For example, a chemistry question might implicitly refer to a chemical reaction by mentioning only the *Mix* or *Combine* event that causes the reaction, as in the following question from a college chemistry text book [10]:

(2) Question 3.9: When the metallic element sodium is combined with the nonmetallic element bromine, $Br_2(l)$, how can you determine the chemical formula of the product?

The question should be paraphrased into the following:

(3) Question 3.9: When the metallic element sodium combines with the nonmetallic element bromine, $Br_2(l)$, how can you determine the chemical formula of the product *of the reaction caused by the combination*?

The example illustrates that naïve encodings, while faithful to original English expressions, may refer to an effect (e.g. the chemical reaction) by its cause (e.g. the combination of chemicals). To form a correct encoding, one needs to identify the causal factors which are only implicit in the text.

From the perspective of a knowledge engineer, it is best to avoid this conflation by cleanly separating the effects of an action from their causes – e.g. by separating the effect of a chemical reaction from the Mix or Combine action that caused it. Therefore, naïve encodings, like the one above, are often misaligned with the ontology of the knowledge-based system to which they are presented, and are properly labeled loose speak.

This loose speak type is similar to INSTITUTION-FOR-PEOPLE-RESPONSIBLE metonymy because both are related to causal factors. It is also similar to PRODUCER-FOR-PRODUCT metonymy because both conflate the causes and effects of an action.

3.2.3. Aggregate

An aggregate is a set of similar objects. An aggregate type of loose speak occurs when one refers to an aggregate by one of its members or *vice versa*. For example, in chemistry, a single molecule is different from a chemical, which is a set of similar molecules. A chemical has properties, such as *state* (solid, liquid, or gaseous) and *solubility*, that a single molecule does not have.

Fig. 1(a) contains an example of the aggregate type of loose speak: a single molecule (*HCl*) does not act as the solute of a solution. It should be replaced by a *Chemical* whose basic structural unit is an *HCl* molecule.

For a naïve user, the distinction between an object and an aggregate is often blurred. For example, a chemist uses context to easily disambiguate the mention of an HCl as referring to a single molecule or a substance comprised of HCl molecules.

For a knowledge engineer, this distinction is crucial. If there is no difference between an object and an aggregate, then the knowledge base contains invalid assertions. For example, if there is no difference between a molecule and a chemical, then the knowledge base may allow an invalid axiom asserting that a single H_2O molecule is in the *liquid* state at room temperature. In order to correct this type of loose speak, the Loose-Speak Interpreter needs to determine which properties

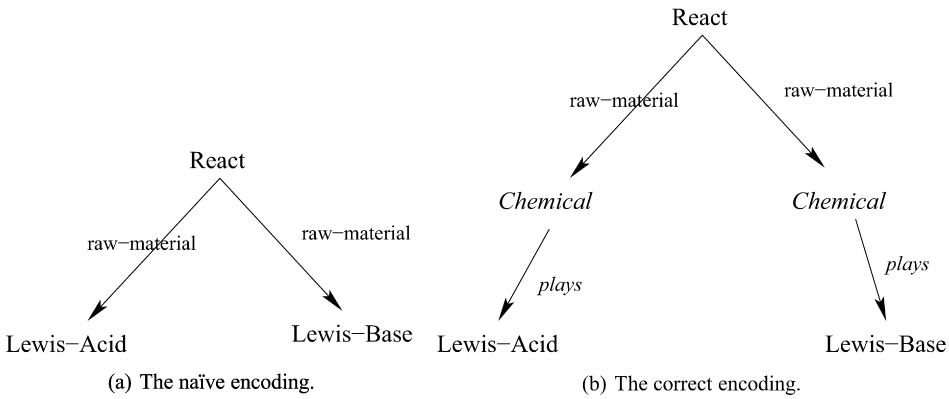


Fig. 2. The naïve and correct encoding of "a Lewis acid reacts with a Lewis base". Notice that the generic *Chemical* concept is added because Lewis acid is a role played by a chemical.

pertain to individual objects and which pertain to aggregates, and to properly replace references of objects with references to aggregates, or *vice versa*.

Although this type of loose speak is similar to PART-FOR-WHOLE metonymy, it differs in terms of the partonomy types to which it applies. The PART-FOR-WHOLE metonymic relation usually applies to heterogeneous sets (e.g., an entity can have different types of parts, such as the various parts of a car), but the aggregate metonymic relation usually applies to sets that are homogeneous (e.g., each individual in the set is a basic unit of the set, such as a single soldier in an army).

3.2.4. Role

A role type of loose speak occurs when one uses a role interchangeably with the entity that plays that role. Knowledge engineers distinguish between roles and entities because they differ in their permanence. Roles exist in the context of events, but unlike entities, they do not exist apart from those events. For example, *Employee* is a role because it exists only during an *Employment* event. In contrast, the *Person* who plays the Employee role persists with or without Employment, and is therefore an Entity. Because of this distinction, roles and entities need to be represented differently to reason with each appropriately [21]. For example, the chemistry knowledge base used in Project Halo contains the role *Lewis Acid* and the entity *Chemical* (a Lewis acid is a role played by a chemical that loses electrons in a reaction). Chemists often use these terms interchangeably, and as a result of such ambiguous usage, role type of loose speak occurs frequently. Fig. 2 shows another example of it.

3.2.5. Overly generic

Sometimes a general concept is used naïvely to refer to a specific concept. For example in chemistry, *Reaction* is often used to refer to a specific type of reaction, such as *Equilibrium-Reaction*. This type of loose speak occurs because it is natural to use different terms to reference the same concept in a discourse. One common type of coreference is to use the superclass of a concept to denote the class itself. For example, "the car" is used to denote a "sedan" mentioned in the discourse. For a naïve agent interacting with a knowledge-based system, it is natural to encode knowledge in a similar fashion.

From the perspective of a knowledge engineer, the distinction between a specific concept and its superclasses is an integral part of the knowledge base. If an overly generic class is used, then properties that belong to the specific class will not be available. For example, the axioms about how to compute *equilibrium constant* are only available to *Equilibrium-Reaction*, not the more generic class *Reaction*. The missing properties may cause the reasoning mechanism of a knowledge base to fail. In order to allow naïve users to naturally encode knowledge and to ensure the correctness of the reasoning process, a Loose-Speak Interpreter must replace the overly generic concept with the correct subclass.

3.2.6. Noun compound

A noun compound is a sequence of nouns composed of a head noun and one or more modifiers. The head noun determines the type of the whole compound (with few exceptions), and the modifiers specialize the type from the head noun. A noun compound type of loose speak occurs when an agent uses a pair of nouns without specifying the semantic relations between them. This type of loose speak occurs frequently because it is natural and efficient to use noun compounds in natural language.

Some noun compounds' semantic relations are single links. For example, in the compound *car engine*, the implicit relation is *part-of*. Other noun compounds require a long semantic path to relate the head noun and its modifiers. For example, the correct interpretation of *animal virus* in the knowledge base built for the Rapid Knowledge Formation project [6] is "a virus that is the agent of an invade, in which the object of the invasion is a cell that is the basic structural unit of an animal". Finding this interpretation involves a hidden predicate (*invade*) and multiple semantic relations (*agent*, *object*, *is-basic-structural-unit-of*).

As these examples illustrate, the interpretation of noun compounds is idiosyncratic. For each noun compound, there are dozens of possible semantic relations to connect the head noun with the modifier [4]; an interpretation of the noun compound is correct only in the context of a specific knowledge base. Nevertheless, noun compounds in naïve encodings must be interpreted because such ambiguities are not tolerated by most reasoning mechanisms.

3.3. Other types of loose speak

In addition to the most frequently occurring types described in the previous section, there are other types of loose speak that are much less common. For example, in addition to overly generic type of loose speak, there is also an overly specific type of loose speak. For example, the specific type *Burn* is used in *cells derive useful energy from "burning" glucose* (*Burn* is a specific type of *Oxidation* in which the reaction is rapid and releases heat and light). In this case, the relatively general concept *Oxidation* is more appropriate than the more specific concept *Burn*. This type of loose speak occurred only once in our study.

3.4. How different types of loose speak are related

Although these six types of loose speak occur frequently, they differ in three ways. First, they differ in terms of the rationale behind the design of the ontology of the knowledge-based system that receives the loose speak. For example, ontology designers separate cause and effect for the sake of cleaner conceptualization, and they distinguish between roles and entities for sound reasoning, but they design partonomies (the designation of parts) in an arbitrary manner. None of these design decisions would be obvious to a naïve user, and it is unreasonable to expect that users' encodings of input would align with the idiosyncratic ontologies.

Second, the types of loose speak differ in terms of why naïve encodings are used. A user may encode naïvely because it is closer to natural language discourse, as is the case for the *overly generic* type of loose speak. A user may also encode naïvely for efficiency, as is the case for *noun compound* type of loose speak.

Third, the types of loose speak differ in terms of the type of semantic relations needed for interpretation. The different relation types include *has-part*, *causes*, *subclasses*, etc. The variety in relation types makes it more difficult for an interpreter to find the correct encoding of a naïve user's encodings.

Despite the differences, these types of loose speak share one important feature: contiguity.[1] For all of these types, a naïve encoding is only slightly different from the correct encoding. In fact, when viewed independently of any knowledge base idiosyncrasies, the naïve encodings are quite reasonable to convey the intended semantics. When in the context of a knowledge base and its idiosyncrasies, the naïve encoding is only a few modifications away from the correct encodings. The types of modifications are numerous, but the total number of modifications needed for a given naïve encoding is limited.

4. Loose-speak interpreter

In the previous section we presented a frequency study, in which we identified frequently used types of loose speak and described them in detail. Despite the differences among these types, we have developed one algorithm that can effectively interpret all of them.

4.1. Input and output

Before we introduce the algorithm of the Loose-Speak Interpreter, we define its task by its input and output. The input to the interpreter is a set of knowledge structures, represented formally. The input knowledge often comes from sentences in natural language, such as example 1 in Section 2.1, which are translated into a formal language by a naïve agent, i.e. a human or a natural language processing system that encodes naïvely, without regard for the ontology of the knowledge base to which it will be applied. The formal language is a logic-based symbolic representation language – in our case, KM [57], a frame-based language with clear first-order logic semantics. There are two basic types of KM constructs: assertions and queries. Additionally, KM also provides a variety of representational structures, such as conditionals, aggregates and universal quantifiers. To avoid issues of KM syntax, we will illustrate our solution using conceptual graphs [61].

The interpreter's output is a corrected encoding of the input. It is written in the same representational language as the input, and it is supposed to convey the intended meaning of the agent's naïve encoding. However, the output is guaranteed to obey the ontological constraints of the knowledge-based system. Therefore, the agent can apply this correct encoding to the system without concern.

Although our interpreter is developed to read and write expressions in the KM language, the algorithm does not depend on KM. We believe it can be adapted to apply to other formal languages that have similar constructs for assertions and queries.

[1] The notion of "contiguity" [36] in linguistics has been used mostly for linguistic metonymy, and we borrowed it for all types of loose speak.

```
Given a triple of the form <C1, r, C2>:
if <C1, r, C2> is a query and there is an answer for the query
   then
      return true
   else
      if exists a triple <C1', r, C2'> in the knowledge base such that
         C1' subsumes or is subsumed by C1 and
         C2' subsumes or is subsumed by C2
      then
         return true
      else
         return false
```

Fig. 3. The resemblance check for a triple. It is based on the hypothesis that if the input does not resemble any existing knowledge, then it may contain loose speak.

4.2. Input decomposition

In order to handle an input of arbitrary length and complexity, the interpreter needs to decompose it into simple and uniform constructs and to process them one at a time.

Our algorithm decomposes an input into a set of binary predicates (triples) in the form of $\langle C_1, r, C_2\rangle$ where C_1 and C_2 are classes or instances of classes and r is a relation. Queries are decomposed into the form of $\langle C_1, r, ?\rangle$ where ? is the value to be queried.[2] A triple corresponds to a labeled edge in a conceptual graph, in which the edge label represents r, and the two vertices represent C_1 and C_2.

4.3. Test-and-repair

After an input is decomposed into triples, the Loose-Speak Interpreter tests each triple for loose speak, and repairs it if an occurrence is found. The Interpreter's test determines whether the triple contains loose speak. In order to tell for certain whether an input contains loose speak, one needs to compare the naïve encodings with the correct encodings. Unfortunately, the correct encodings are unknown to the interpreter, so the test has to use a heuristic function to detect loose speak. Our interpreter uses the following:

> If the triple violates structural constraints, then it must contain loose speak (because correct encodings are consistent with the structure of the knowledge base).

The only structural constraints used by the interpreter are based on the domain and range of the relation in a triple. The constraint violation reflects the common approach of linguistic metonymy detection based on *selectional restriction violations*. In the field of linguistic metonymy resolution, a metonymy is usually recognized by a violation of the semantic restrictions that predicates impose on their arguments. Specifically, for a triple $\langle C_1, r, C_2\rangle$, the domain of r must subsume C_1, and the range of r must subsume C_2.

Although the constraint check returns true positives, it may also return false negatives. Some researchers [24,46,55] have pointed out that selectional restriction violation (or constraint check) is not sufficient for all occurrences of metonymy. For our interpreter, the constraint check is complemented by a resemblance check. Fig. 3 shows the algorithm. The resemblance check is based on the hypothesis that if the input does not resemble any existing knowledge, then it may contain loose speak, because studies have shown that "one frequently repeats similar versions of general theories" [14]. If an input does not resemble any previously entered knowledge, then it is highly unusual, and it deserves a closer look by the repair routine. The knowledge resemblance test may return many false positives (i.e. erroneously classifying a correct encoding as containing loose speak) since non-resemblance does not prove misalignments, but it returns many true negatives (i.e. correctly classifying a correct encoding as not containing loose speak) as well. The knowledge resemblance test complements the constraint check by ensuring a triple that is judged not to contain loose speak is truly free of loose speak.

The resemblance test, applied to a triple, is implemented as follows. If the triple represents a query (i.e., the third element of the triple is *?*), the triple passes the test only if the knowledge base can compute one or more fillers for the

[2] We assume a single query triple per question. If a question asks for the value of multiple triples, it can be converted into multiple questions with a single query triple each.

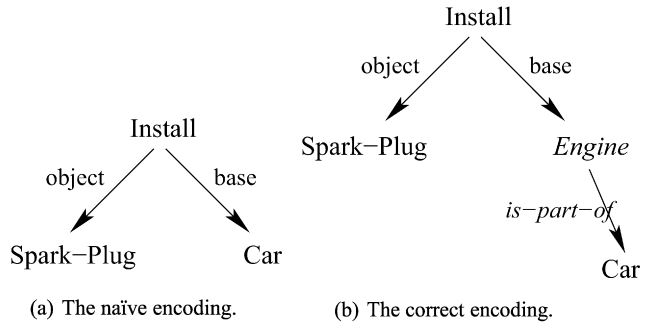


Fig. 4. An example in which the resemblance test fails to detect loose speak. If the knowledge base has a prior example of *Device* being the object of an *Install* and *Spark-Plug* is a subclass of *Device*, then the naïve encoding will pass the resemblance test.

```
Given a triple of the form <C1, r, C2>
   DEPTH = 0
   RESULT = nil
   While DEPTH < MAX-DEPTH and not(RESULT)
      RESULT = search_head(<C1, r, C2>)
      If not(RESULT) then
         RESULT = search_tail(<C1, r, C2>)
         If not(RESULT) then
            RESULT = bi_dir_search(<C1, r, C2>)
         end if
      end if
      C1 = get_subclasses(C1)
   end while
   return RESULT
```

Fig. 5. The `specialize_head` procedure. It specializes C_1 of the triple $\langle C_1, r, C_r \rangle$ into its subclasses, and calls the `search_head`, `search_tail` and `bi_dir_search` procedures.

tail. (This is the basic question-answering function of a knowledge base.) Otherwise, the triple $\langle C_1, r, C_2 \rangle$ passes the test only if the knowledge base contains a triple $\langle C_1^{'}, r, C_2^{'} \rangle$, such that $C_1^{'}$ subsumes or is subsumed by C_1 and $C_2^{'}$ subsumes or is subsumed by C_2.

Although the resemblance test reduces the number of false negatives, it cannot catch all cases of loose speak. For example, if given the input "install the spark plug in the car", a naïve encoding may look like Fig. 4(a). The *Spark-Plug* and *Car* concepts may satisfy the type constraint of *Install*, and the resemblance test may pass because the knowledge base has a prior example of installing a device in a car and *Spark-Plug* is a subclass of *Device*. However, if a *Spark-Plug* is part of an *Engine* instead of a *Car*, then the correct encoding would be like Fig. 4(b), and the loose speak occurrence would elude the resemblance test.

If the interpreter detects loose speak in a triple, it repairs the triple by searching the space of similar triples for one that is correct. The search is conducted through two specialization procedures. The specialization steps search the subclasses of C_1 or C_2 to see if they are related through the relation r. As shown in Fig. 5, the `specialize_head` procedure is made of three searches: `search_head`, `search_tail`, and `bi_dir_search`. `Specialize_tail` is identical to `specialize_head` except that it specializes C_2 instead of C_1.

The `search_head` and `search_tail` procedures perform breadth-first searches. Given the triple $\langle C_1, r, C_2 \rangle$, the `search_head` function starts at C_1, traverses all semantic relations, such as *has-part*, *agent*, and *subclasses*, and stops when a suitable instance is found or a search depth limit (`MAX_DEPTH`) is reached. An instance C_3 is suitable if it is a superclass or a subclass of C_2 and the path from C_1 to C_3 contains the relation r. The successful search path is returned as the interpretation for the input. Fig. 6 gives an example of `search_head`.

To avoid unintended interpretations, the set of semantic relations traversed by the search does not include the *superclasses* relation. If both *subclasses* and *superclasses* are included in the search path, then any concept can be found from C_1

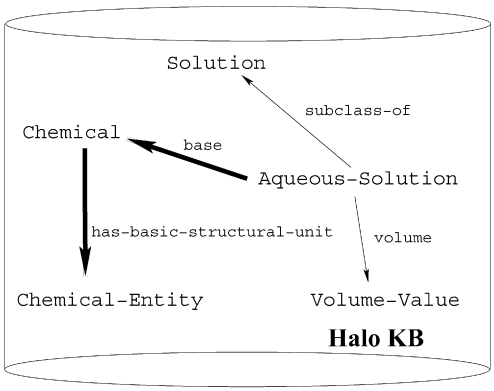


Fig. 6. The application of the "search-head" function on the triple ⟨*Aqueous-Solution, base, HCl*⟩. The function is called because this triple does not resemble any existing path in the knowledge base. The bold lines in the above figure show how the result is found.

by climbing up and down the taxonomy, and a large number of spurious interpretations might be returned for the given input.

If multiple suitable instances are found, then the algorithm ranks them and selects the most appropriate one. The ranking scheme is based on the Occam's Razor heuristic, which prefers simple answers. The complexity is measured by the length of the path from C_1 to a found instance: a shorter path is ranked higher than a longer one. Shorter paths are considered simpler because it is easier to find the suitable instance from C_1. Preferences are given to paths containing essential relations as determined by valency theory [60]. Valency (or valence) refers to the arguments of a verb, and the relations between the verb and its arguments, such as *object* and *agent*, are the essential relations. When computing the length of a path, non-essential relations are considered to be of length (cost) 1.0, while essential relations are only 0.55.

The Loose-Speak Interpreter has two modes for selecting the answer. The automatic mode always chooses the shortest path as the answer; the manual mode presents all the answers sorted by path length and prompts the user to select one.

The `search_tail` function is implemented similarly, except that its search starts at C_2. The `bi_dir_search` starts from both C_1 and C_2.

The search is conducted in a breadth-first manner for two reasons. First, because each occurrence of loose speak is a naïve encoding that has a limited number of differences from its correct encoding, a correct interpretation should be closely related to the naïve input. A very deep search will only return encodings that are not closely related to the input. If only a shallow search is needed, then a brute force search such as breadth-first is sufficient. The maximum search depth in our system is set to four based on empirical study on a small pilot data set.[3] Second, a breadth-first search returns the shallowest interpretation without examining the whole search space.

If no interpretation is found when the depth cutoff is reached, the interpreter returns the original encoding. In addition, if loose speak was detected by a constraint violation, the interpreter returns an error report.

This algorithm applies to all commonly used loose speak types. Metonymy, role, causal factor, aggregate and overly generic types of loose speak are represented as labeled triples, and `specialize_head` and `specialize_tail` are applied if a triple fails either the constraint test or the resemblance test. A noun compound is represented as an unlabeled triple of the form: ⟨*ModifierClass*, *nil*, *HeadClass*⟩. Because there are no unlabeled triples in the knowledge base, noun compounds will fail the resemblance test, and consequently will trigger `search_head` and `search_tail`.

5. Evaluation

In this section, we present the results of various evaluations of the Loose-Speak Interpreter. There are two questions we want to answer through the evaluation. First, we want to know whether the interpreter reduces the brittleness of knowledge-based systems caused by input that misaligns with the ontology of the knowledge base. We evaluated this by applying the interpreter to many naïve encodings and measuring its performance in correcting them (Section 5.1). Second, we want to know the knowledge requirements of the Loose-Speak Interpreter (Section 5.2). If it requires special knowledge, such as detailed knowledge about the input, then any gains in knowledge base interaction by the Loose-Speak Interpreter could be overshadowed by the efforts needed to formulate the additional knowledge. In that case, loose-speak interpretation would be of little net value.

[3] Because different the knowledge bases have different sizes and granularities, the maximum search depth limit may vary when other knowledge bases are used.

5.1. How well does the Loose-Speak Interpreter perform?

To evaluate the effectiveness of the Loose-Speak Interpreter in knowledge base interaction, we conducted three experiments: a user study (Section 5.1.1), a noun compound interpretation study (Section 5.1.2), and a study of the interpretation of loose speak in natural language texts (Section 5.1.3).

5.1.1. User study

First, we asked several users to encode a set of questions, then measured the Loose-Speak Interpreter's ability to test and repair the naïve encodings.

5.1.1.1. Setup For this experiment we used a set of fifty multiple choice questions from various mock Advanced Placement chemistry tests. This set was compiled by a chemist (anonymous to us), who also provided the answers for the questions. This data set was distinct from the two sets of questions used in the frequency study in Section 3.1, and it was distinct from the training data that we used to refine the interpreter.

We recruited three users, with varying backgrounds in knowledge engineering and chemistry, to pose these questions to the chemistry knowledge base built for Project Halo [66]. All three were familiar with KM [57], the knowledge representation language used in the experiment, but none was familiar with the chemistry knowledge base and its idiosyncrasies. Their knowledge engineering experience varied from a few months to many years. Their chemistry backgrounds varied from weak (high school chemistry decades ago) to medium (college chemistry years ago).

The users were given a short (three page) tutorial on encoding questions. It contained two sections: format and vocabulary. The format section explained that question encodings should have a context part, which gave the contextual information of a question, and an output part, which specified the value being sought. The format section illustrated this with three examples. The vocabulary section listed commonly used classes and slots in the chemistry knowledge base and their corresponding chemistry terms. Users were also referred to a web page containing the complete listing of slots and classes; they were allowed to take as much time as necessary to complete the questions. These brief instructions were not a complete tutorial on using the knowledge base, as evidenced by the high rate of loose speak in the users' encodings of questions.

We used metrics of precision and recall to evaluate the interpreter's performance. Precision and recall were defined as follows [34]:

$$Precision = \frac{\#\ of\ correct\ answers\ given\ by\ system}{\#\ of\ answers\ given\ by\ system}$$

$$Recall = \frac{\#\ of\ correct\ answers\ given\ by\ system}{total\ \#\ of\ possible\ correct\ answers}$$

For our experiment, the number of correct answers was the number of question encodings that were interpreted correctly. The number of answers given by the system was the number of question encodings for which the interpreter detected loose speak and found an interpretation. This includes three types of encodings: first, naïve encodings that are correctly detected and interpreted; second, naïve encodings that are correctly detected but erroneously interpreted; and third, correct encodings that are erroneously detected as containing loose speak. Finally, the number of all possible correct answers was the number of all the question encodings that contained loose speak. An interpretation was determined to be correct if it completely aligned with the chemistry knowledge base and conveyed the intended meaning of the question encoding as judged by us. Precision estimated the likelihood of a correct interpretation when an interpretation was found; recall was a measurement of coverage.

Our experiment did not include the noun compound type of loose speak because chemistry is an unrepresentative domain for evaluating noun compound interpretation. Although noun compounds, such as "carbon monoxide" (a molecule containing both an oxide and a carbon), occur frequently in chemistry questions, their interpretations all are of the same format, and they do not require inferring a variety of semantic relations. The effectiveness of the interpreter on noun compounds was evaluated separately, as described in Section 5.1.2.

For this experiment, we ran the interpreter in "batch mode". If the interpreter found multiple interpretations for any question, we treated the first interpretation as its sole response.

Table 2
The users' background and performance at encoding questions.

	KE experience	Chemistry knowledge	Experiment length	Questions encoded	Correct encodings
User A	0.5 yr	medium	10.5 hrs	47 (94%)	3 (6.4%)
User B	17 yrs	weak	9.95 hrs	48 (96%)	4 (8.3%)
User C	0.25 yr	medium	10 hrs	44 (88%)	5 (11.4%)
Average	6 yrs	medium	10.15 hrs	46.3 (92.7%)	4 (8.7%)

Table 3
The performance of the Loose-Speak Interpreter at interpreting users' naïve encodings.

	Precision	Recall
User A	97.4%	90.5%
User B	100%	90.7%
User C	96.8%	87.8%
Average	98.1%	89.7%

5.1.1.2. Data analysis and discussion The experimental results are shown in Table 2 and Table 3. The first two columns of Table 2 give the users' background in knowledge engineering and in chemistry; the next three columns show how well the users performed by recording the time it took them to encode the test questions, the number of questions they were able to encode, and the percentages of correct encodings. Table 3 shows how well the interpreter performed.

Based on the data, we can draw three conclusions:

(1) Loose speak is very common – which confirmed the results of our preliminary study (described in Section 3). Less than 9% of the encodings on average contain no loose speak. This underscores the importance of the Loose-Speak Interpreter for building useful knowledge-based question-answering systems.
(2) The Loose-Speak Interpreter works well. As shown in Table 3, the precision of the interpreter is above 95%, and the recall is around 90%. This shows that the search algorithm we used is suitable for interpreting naïve encodings. The high recall indicates that most occurrences of loose speak were correctly detected and interpreted, hence the interpreter was of great assistance in reducing the brittleness of the knowledge-based system. The high precision indicates that the interpreter had only a few false positives, hence the interpreter posed little overhead to the knowledge base interaction. In addition we found that 54% of the loose speak occurrences were detected by constraint violations, and the rest by the resemblance check. This suggests both checks are important for detecting loose speak.
(3) The Loose-Speak Interpreter is helpful in assisting novice users in posing questions. The results show that there is no clear correlation between the users' knowledge engineering experience and the number of questions they are able to encode. Users with little knowledge engineering experience were able to encode large percentages of questions quickly. This is good news; it suggests that with the burden of loose-speak interpretation handled automatically, users can focus on the content of questions instead of the details of knowledge engineering.

5.1.2. Noun compound interpretation

Because the Project Halo data set contains relatively few types of noun compounds, we evaluated our Interpreter's performance on that type of loose speak separately. As described in Section 3.2.6, noun compounds contribute to the ambiguity of natural language. They have received considerable attention from researchers who have contributed many methods for interpreting them. One strength of our approach is that it interprets noun compounds using the same algorithm that is used for resolving other types of loose speak. Our algorithm treats a noun compound as a binary predicate with an unspecified relation, and it searches the knowledge base for the best relation between the head noun and its modifier.

5.1.2.1. Setup To avoid getting results that are skewed to a particular domain or knowledge representation, we used a variety of quite different data sets. The first consists of 224 noun compounds from a college-level cell biology text [1]. The second consists of 294 noun compounds from a small-engine repair manual [3]. The third data set consists of 224 compounds from a Sun Sparcstation manual [12]. The nouns used in these data sets are manually mapped to the corresponding concepts in knowledge bases on these topics.

The interpretation of a noun compound is considered correct if the first path returned by the algorithm provides a *sensible* interpretation, as judged by a human oracle. If multiple paths of the same lengths are returned, then one path is randomly selected as the first path. Here is an example of one such *sensible* interpretation. Given "computer expert", the interpretation "a computer acting as an expert" is correct, even though another interpretation ("an expert on computers") might be the first to come to mind. This criterion is used for two reasons. First, when data sets are extracted by other researchers from text as pairs of nouns, the associated context is lost, and it is hard to know the "right" interpretation without any context. Second, this criterion is used in previous approaches such as [7,65]. Using the same criterion makes it easier to compare the results with other reported studies.

The metrics we used to evaluate performance are precision and recall as defined in Section 5.1.1.1.

5.1.2.2. Knowledge bases The knowledge bases used for our experiments consist of a set of concepts and a set of axioms for each one. The concepts are related to one another through subsumption relations to form a hierarchy. The axioms on one concept are assertions of semantic relations between that concept and other concepts. For example, our knowledge base defines *Action* as a concept about things that *happen*. The axioms on Action include, for example, "every Action acts on – and changes – some Entity", where *Entity* is defined as a concept that describes things that *are*. Axioms are either local or inherited from superclasses.

Table 4
The results of evaluating the Loose-Speak Interpreter on noun compounds. Each data set contains more than 200 pairs of nouns, and the interpreter's precision and recall are around 80% across all three data sets. The performance is similar to that achieved by previous systems.

	Data size (N)	Precision	Recall
Biology	224	93.8%	93.8%
Engine	294	85.2%	74.5%
Sparc	224	84.5%	73.2%
Average	247.3	87.8%	80.5%

Despite these commonalities, the knowledge bases differ significantly. They differ in terms of how they were built. The knowledge base for the biology text was built using the generic Component Library [6] to answer end-of-the-chapter style questions, as one of the challenge problems for DARPA's Rapid Knowledge Formation project [59]. The knowledge bases for the other two data sets (the small-engine repair manual and the Sparcstation manual) were built "on top of" the knowledge in WordNet [25], which has more than 100,000 concepts. We augmented WordNet with the upper ontology of the generic Component Library plus about ten concepts that are important to each of the two domains (whose partonomies are not complete in WordNet). Through this process, we encoded 416 concepts in about 50 man-hours. The advantage of using WordNet as the foundation for these knowledge bases is two-fold: it includes most of the terms used in the data sets, linked with both taxonomic and partonomic relations, and it is widely available and well used. The knowledge bases also differ in terms of content. Other than the shared upper ontology of the generic Component Library, they have few concepts in common.

5.1.2.3. Interpreter performance Table 4 shows the performance of the interpreter for the three data sets in terms of precision and recall. Both precision and recall are around 80% across all three knowledge bases. This is similar to the performance achieved by previous noun compound interpretation systems described in Section 6.2.

We have also measured the distribution of the path lengths of the top interpretations from the data set. The distribution (Fig. 7) shows the length of a typical interpretation found by the Loose-Speak Interpreter. If the interpretation of a noun compound is a subsumption relation, for example "an oak tree" is interpreted as "a tree" or "an oak", then the interpreter's spreading activation starts and stops at "oak" because "oak" is a kind of "tree". Since the spreading activation does not traverse through any other relations, the path length is zero. If the interpretation of a noun compound is made of more than one relations – for example "animal virus" is interpreted as "a virus that is the agent of a invade, whose object is a cell that is the basic structural unit of an animal" – then the path length is greater than one. This data set does not include cases when no interpretation is found since the focus of the study is on the distribution of the interpretation path length.

The distribution shows two important points. First, a correct interpretation of about 90% of the noun compounds involves zero or one semantic relations. This means that the spreading activation search conducted by the Loose-Speak Interpreter is usually very shallow. Therefore, even if the knowledge base is very large, the search can be conducted quickly. Second, a correct interpretation of the remaining 10% of noun compounds requires multiple relations. Methods that rely on classifiers [39] will not be able to interpret these noun compounds correctly, and knowledge base search seems to be required.

5.1.3. Interpretation of natural language input

The user study in Section 5.1.1 was based on the encodings of human users. We also evaluated the Loose-Speak Interpreter on encodings produced by a natural language processing program. This was conducted to compare the interpreter's performance on human encodings with performance on encodings generated automatically. The comparison is needed because the loose-speak occurrences in human encodings may be significantly different (probably worse) than those in machine encodings.

5.1.3.1. Setup The natural language processing program we used was an interpreter for the Computer Processable Language (CPL) [13], which interprets sentences written in a simplified version of English. A basic CPL sentence is of the form "subject + verb + complements + adjuncts", where complements are elements needed to complete a sentence and adjuncts are modifiers. CPL was used in phase II of Project Halo to interpret questions in physics, chemistry, and biology. The questions were written in CPL by domain experts who did not know the ontology of the knowledge base that would be used to answer the questions (because the knowledge bases were built by different domain experts).

The Loose-Speak Interpreter was used in conjunction with the CPL interpreter in two ways. First, the Loose-Speak Interpreter was used as a noun compound interpreter whenever a noun compound was encountered in a sentence. Because CPL did not allow noun sequences longer than two, the CPL interpreter did not need to bracket noun compounds [5]. Second, after the CPL interpreter had created a formal representation of a sentence, the Loose-Speak Interpreter was called to detect and resolve any remaining occurrences of loose speak. Notice that the input of the Loose-Speak Interpreter in this evaluation came from the encoding of single sentences, hence discourse effects did not need to be handled.

5.1.3.2. Knowledge bases and users Six knowledge bases were used in this experiment covering three domains: biology, chemistry and physics. Unlike the chemistry knowledge base used in the study in Section 5.1.1, these knowledge bases

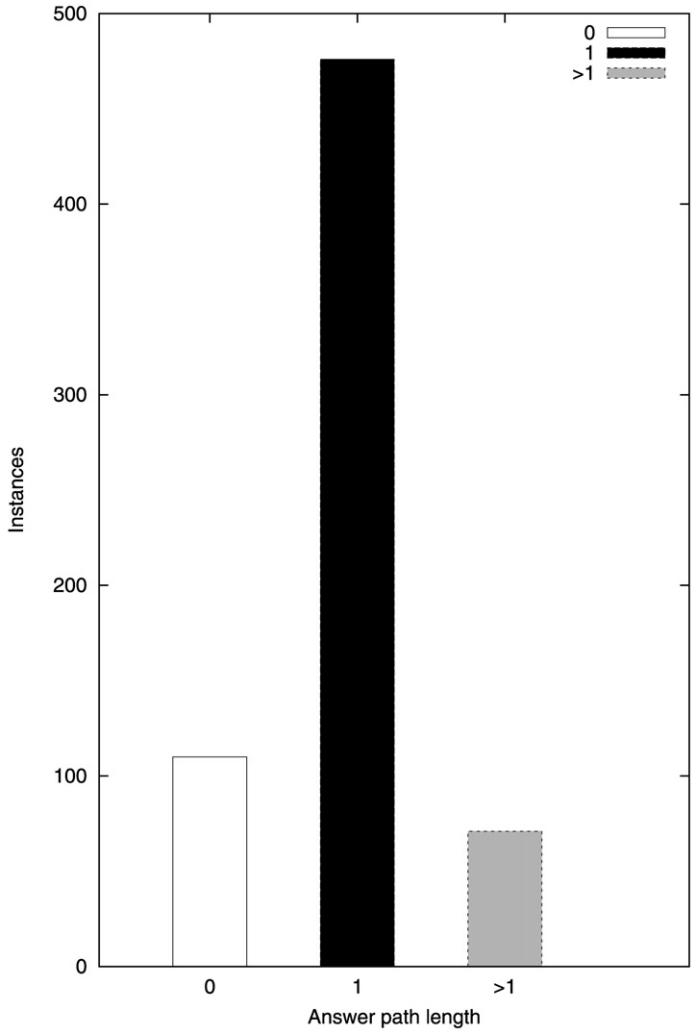


Fig. 7. The distribution of path lengths for the top interpretations.

were not built by knowledge engineers; instead they were encoded by six domain experts (two per domain). Because domain experts have little training in knowledge engineering, we expect that the knowledge bases they build will have more gaps and errors.

The test questions were based on Advanced Placement test questions, and they were posed by five different users. Four of these users were undergraduate students, and one was a high school student. None of them had experience with knowledge-based systems.

A typical question and its CPL formulation looked like this:

Original Question A cyclist must stop his bike in 10 m. He is traveling at a velocity of 17 m/s. The combined mass of the cyclist and bicycle is 80 kg. What is the force required to stop the bike in this distance?

CPL Formulation An object moves. The mass of the object is 80 kg. The initial velocity of the object is 17 m/s. The final velocity of the object is 0 m/s. The distance of the move is 10 m. What is the force on the object?

The correctness of the responses by the interpreter was decided by a separate panel of five judges, each responsible for the questions posed by one user. The performance of the interpreter was measured by precision and recall as defined in Section 5.1.1. For this experiment, we ran the interpreter in "batch mode" as before. If any question had multiple interpretations, only the first one was chosen.

Table 5
The results of evaluating the Loose-Speak Interpreter on encodings produced by a natural language processing program.

		Num of questions	LS noun compounds		LS overall	
			Precision	Recall	Precision	Recall
Biology	user 1	47	100%	28%	100%	100%
Chemistry	user 1	21	71%	14%	83%	33%
	user 2	8	100%	12%	92%	71%
	user 3	21	100%	27%	100%	100%
Physics	user 1	41	92%	87%	82%	78%
Overall average		138	92%	38%	86%	68%

5.1.3.3. Data analysis and discussion Table 5 shows the results of the experiment. The first two columns show the domain and the users; the next column shows the number of questions encoded. The remaining columns show the performance of the Loose-Speak Interpreter using encodings created by the CPL interpreter.

In general, the precision of the Loose-Speak Interpreter at all three tasks is high. On average, its precision for interpreting noun compounds is above 90%, and its overall precision is above 85%.

The recall of the interpreter is lower. On average, recall for the noun compound interpretation task is 38% and overall recall is 68%. An analysis of the failed cases reveals that there are three main causes for the low recall:

(1) Improper knowledge base concept mapping. One of the first steps of the CPL interpreter is mapping word strings to knowledge-base concepts. In the case of a novel word, the CPL interpreter finds the closest concept or creates a new concept based on the input string. Unfortunately, such mappings are not always perfect, for example, the word "count" as in "what is the total count of chromosomes" was mapped to a *Person* because one sense of "count" is "a nobleman". Incorrect word sense disambiguation results in wrong input for the Loose-Speak Interpreter, which causes it to fail to find an interpretation.
(2) Parsing errors. There are domain-specific technical phrases, such as "escape velocity", that are noun compounds but should be recognized by the parser as single terms and not interpreted. However, the parser sometimes fails to recognize them, and it passes these noun compounds to the Loose-Speak Interpreter, which fails to interpret them.
(3) User errors. During the process of rephrasing a question into simplified English, an user may formulate erroneous input to the system. For example, one user rephrased "a screw falls .." into "There is a move. *The move has a fall.*", which is nonsensical.

To quantitatively measure the impact of these three causes, we counted the number of loose speak occurrences that the interpreter failed to interpret because of each one. Of a total of 112 failed instances, 68 are due to the first cause, 13 are due to the second cause and 2 are due to the third cause. If we exclude the failed cases attributed to these three causes, then the average recall for the noun compound interpretation task is 67% and overall recall is 100%. This is similar to the performance of the Loose-Speak Interpreter in the user study and the noun compound interpretation evaluation described in Sections 5.1.1 and 5.1.2.

Compared with the results from Section 5.1.1, the Loose-Speak Interpreter has demonstrated similar precision with lower recall. The difference is mainly due to the difficulties in natural language processing which result in incorrect input to the Loose-Speak Interpreter. The result suggests that the Loose-Speak Interpreter can handle human encodings as well as automatically generated correct encodings.

5.2. What type of knowledge is needed?

One common and justifiable criticism of knowledge-based systems is that they require detailed domain specific knowledge, which is often difficult to obtain. If loose speak can be successfully interpreted without much knowledge, then the problem is avoided: the small amount of knowledge that is required can be easily encoded. We evaluated the knowledge requirements of the Loose-Speak Interpreter by measuring the contribution of each level of the ontology to the performance of the interpreter. If a level is ablated, and the interpreter's performance is significantly impacted, then the contribution from that level is large, otherwise, the contribution is small.

5.2.1. Setup

The challenge in measuring an algorithm's sensitivity to knowledge base content is that the results may vary across domains and across knowledge bases. We attempt to neutralize these factors by replicating our study in three domains with quite different knowledge bases. We used the three noun compound data sets (biology, small-engine repair manual and Sparc workstation owner's manual) and the three associated knowledge bases described in Section 5.1.2.

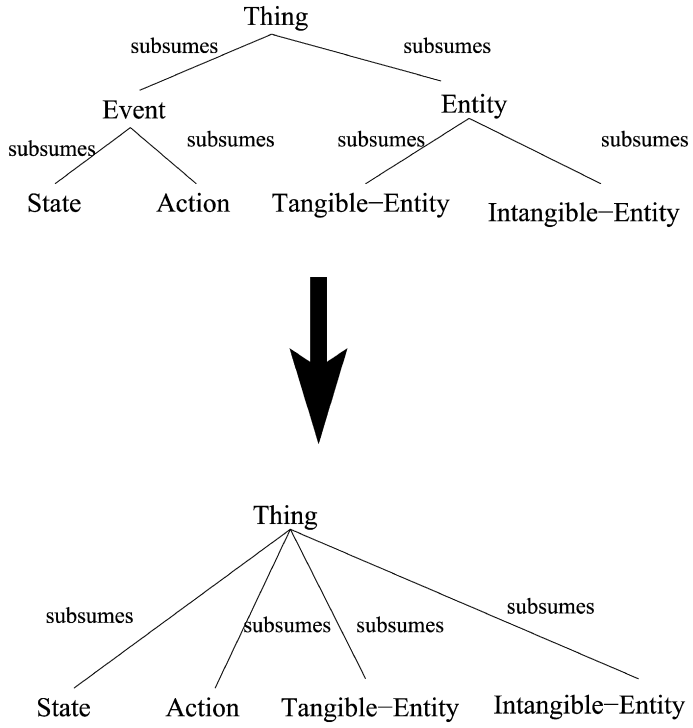


Fig. 8. The ablation of level 1 of a sample taxonomy.

5.2.2. Ablation steps

The sensitivity of the algorithm to each level of the ontology is measured through a series of ablations. When a level is ablated, the concepts on that level and all their axioms are deleted from the knowledge base. If these deleted concepts have subclasses, then the subsumption hierarchy is repaired by connecting the subclasses to the deleted concepts' superclasses. As a special case, when the 0th level (the root level concept) is deleted, it is replaced by a generic concept of *Thing*. Because the root level concept is vacuous, deleting it has no affect. As an example, Fig. 8 illustrates the ablation of level 1.

5.2.3. Data analysis and discussion

In this section, we analyze the experimental data to determine the contribution of each level of the knowledge bases' ontology to the task of interpreting noun compounds and to explain the results.

Fig. 9 shows the effect of ablating different levels of the ontology for the three knowledge bases in terms of precision and recall. Without any ablation, both precision and recall are around 80% across all three knowledge bases.

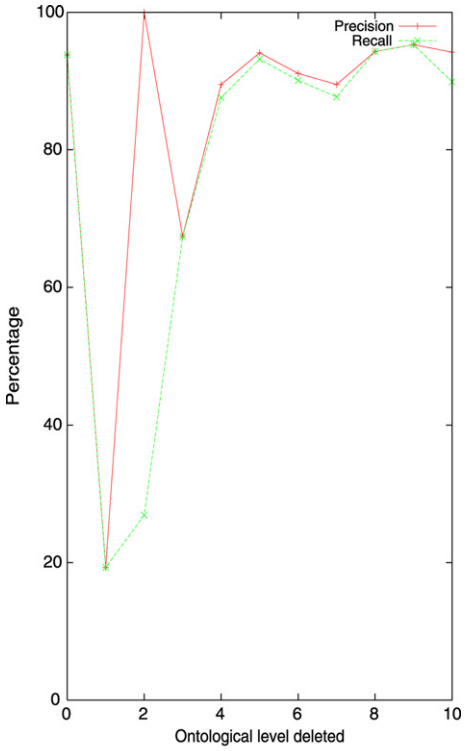
Ablating level one causes a big drop in both precision and recall. Ablating level two introduces a big gap between precision and recall. The gap indicates that the algorithm does not find interpretations for many noun compounds. As lower levels in the ontology are ablated one at a time, the impact diminishes and performance improves to the level of a knowledge base with no ablations.

The contribution of the first two levels of the ontology is observed across all three data sets and knowledge bases. This pattern strongly suggests that the top levels of the ontology are the most important for the noun compound interpretation task.

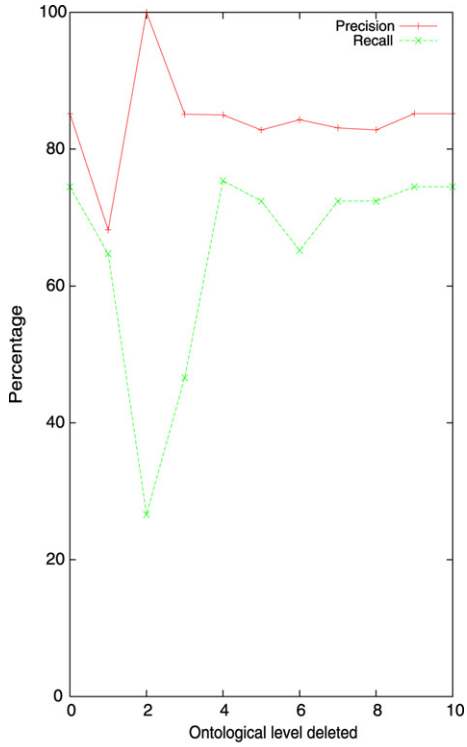
Perhaps the top levels of the ontology are more important than lower levels because they contain many axioms important to the noun compound interpretation task. Ablating these levels has the effect of deleting the associated axioms – not only from the concepts at these levels, but also from all concepts below them in the taxonomy (because the axioms are passed by inheritance from superclasses to subclasses).

To test this hypothesis, we counted the number of axioms associated with concepts at each level of the ontology. There are various ways to count axioms – we chose a simple and conservative one. For this study, we considered an axiom about concept C_1 to be an assertion of a semantic relation between instances of C_1 and instances of another concept C_2. For example, one axiom associated with the concept *Car* is: "every car has 4 wheels". We counted only local (non-inherited) axioms, and we counted in minimum fashion, so this axiom is counted once, not four times.

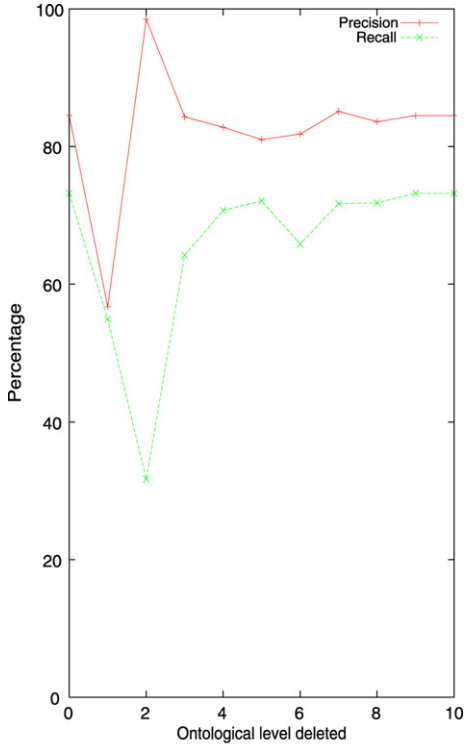
Figs. 10(a), 10(b), and 10(c) compare the impact of ablating different ontological levels (in terms of precision and recall), with the average number of axioms per concept on each level. The data show that upper levels of the ontology contribute the most to task performance, yet they contain relatively few axioms; lower levels of the ontology contribute much less, yet they contain relatively more axioms.



(a) Biology data set.

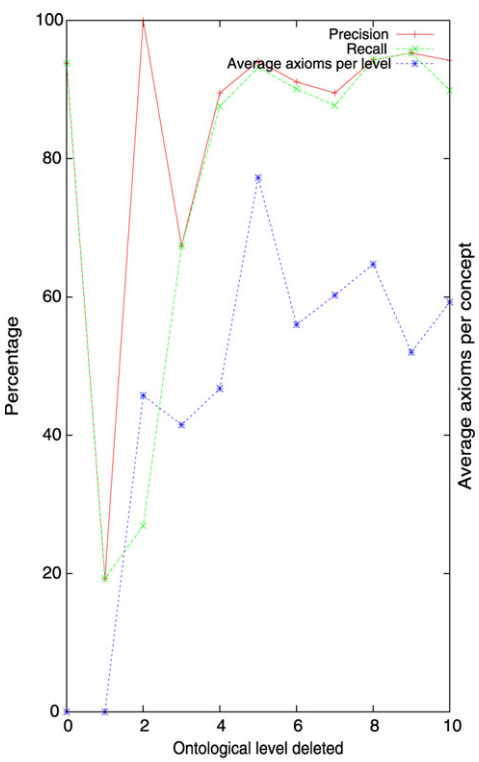


(b) Engine data set.

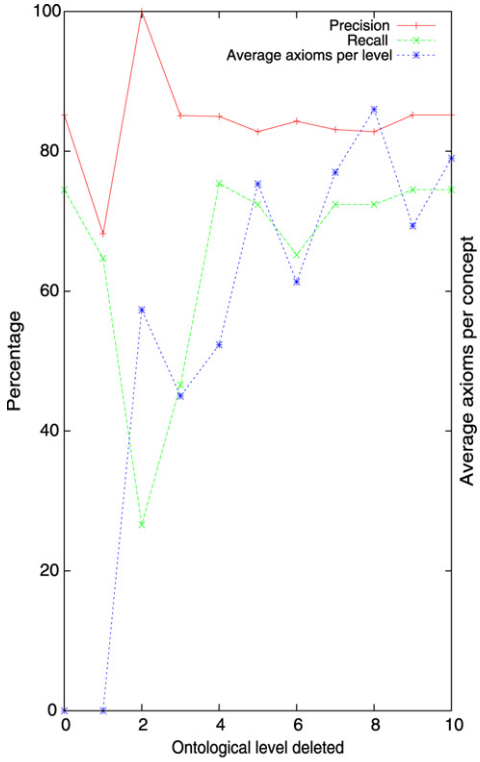


(c) Sparc data set.

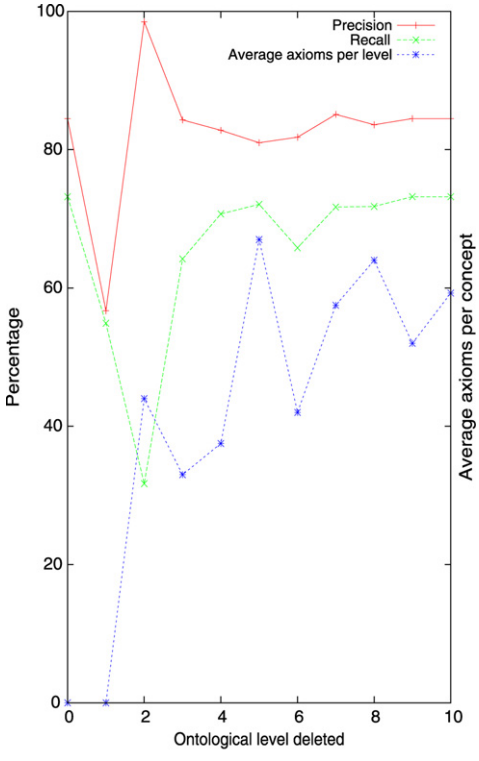
Fig. 9. The impact on precision and recall of ablating different levels of the ontology for the three data sets.



(a) Biology data set.



(b) Engine data set.



(c) Sparc data set.

Fig. 10. Comparison of the impact of ablating different levels of the ontology with the average number of axioms on each level.

We conclude, therefore, that the number of axioms in the top levels of the ontology is not what makes these levels important. Rather, we suggest that they are important for noun compound interpretation for two reasons:

(1) Top levels of the ontology include concepts that make important ontological distinctions. For example, ablating the level that introduces *Entity* and *Event* blurs the distinction between widely different classes of concepts, which causes the search to stop with erroneous results. Consequently, many more interpretations are returned, and because we use only the first interpretation, the probability of it being correct is reduced.
(2) Although they contain relatively few axioms, axioms in top-level concepts are important for the task. The top-level ontology contains the most frequently used axioms, such as the fact that "every Action involves an object that is acted upon". These axioms are used in the search as a step along the way. Deleting these axioms makes it difficult to find an interpretation for many of the noun compounds, thereby causing recall to lag behind precision.

6. Related work

Early work by Hobbs [32] foreshadowed the issues we address in this research. Hobbs observed that people view the world at varying levels of granularity and they shift levels based on their needs. He advocated adding an explicit "grain-size" argument to predicates in a knowledge bases so that the reasoning process could avoid paradoxes caused by assertions at conflicting levels of granularity. Instead of introducing an explicit grain-size argument, which is cumbersome and potentially difficult for both knowledge engineers and users, we present a method that automatically converts expressions to a common granularity.

More recently, researchers have studied related issues, but usually with a focus narrower than ours. For example, many researchers have focused on one type of loose speak, such as metonymy or noun compounds, and proposed methods for resolving it. In the following sections, we take a detailed look at these methods.

6.1. Metonymy

Linguistic metonymy has been studied extensively in computational linguistics because of the high frequency of metonymic expressions in language. Markert and Hahn [46] found metonymic expressions in 15% of the utterances in a German-language corpus of information technology test reports. They found as many as 50% of the 100 randomly drawn occurrences of "*BMW*" were used metonymically.

There are three approaches to linguistic metonymy resolution. First, the rule-based approach [24,29,43,68] contains explicit knowledge about how to interpret some types of metonymy by matching rules with inputs to produce interpretations. This approach is easy to implement because no large knowledge base is required, but it can handle only a fixed number of metonymy types based on the rules.

The second basic approach to linguistic metonymy interpretation involves systems that interpret metonymies based on knowledge base searches [11,33,45]. Unlike the rule-based systems, they do not contain explicit knowledge of different types of metonymy, instead they rely on searching a general-purpose knowledge base. Our interpreter is most similar to this approach. Although this approach has no explicit limit on the types of metonymy it can handle, the requirement of an extensive knowledge base makes it difficult to implement. Our interpreter avoids this cost by operating in a knowledge – acquisition and question – answering setting in which a knowledge base already exists.

Compared to these systems, our interpreter uses a similar search strategy but a different metonymy detection mechanism. In addition to being invoked when a constraint violation is found, our interpreter suspects that the input contains metonymy if there is no prior knowledge similar to it. Although a constraint violation, occurring either within a sentence or within a discourse structure [45], is a more precise indicator of metonymy, the lack of prior knowledge is a more stringent indicator for some cases. For example, if a car is represented as having an engine part and an engine is represented as having a carburetor part, then the query "the carburetor part of a car" contains metonymy even though it does not violate any type constraints (because both *Engine* and *Carburetor* satisfy the type constraint of the *has-part* relation). However, the interpreter will search for an interpretation of the query because there is no prior knowledge about the "carburetor part of a car", and it will find "the carburetor part of the car's engine" as an interpretation of the original input. The prior knowledge will uncover more metonymies than the type mismatch indicator used by these related systems, and this advantage should be reflected in a higher recall value during the evaluation of the interpreter.

The availability of large annotated corpora [48] for linguistic metonymy has given rise to a third approach – classification based linguistic metonymy recognition and resolution [31,47,56]. It uses classification algorithms, such as maximum entropy [23], decision tree, logistic regression [52] etc, on syntactic, lexical and semantic features to learn to recognize and classify different types of metonymy occurrences. The classification based approach has performed well at the metonymy resolution shared task at SemEval 2007 [49] and it does not rely on violations of selectional restriction to recognize metonymy. This allows the recognition and interpretation of metonymy without a large knowledge base. Unfortunately, it is difficult to adopt this approach for loose-speak interpretation because it requires a corpus of examples and little knowledge. This is the opposite of the knowledge – acquisition and question – answering setting, which has few examples but lots of knowledge.

Table 6
Some of the common semantic categories used in noun compound studies in computational linguistics.

Relation name	Example
material	"marble statue"
object-of	"troop movement"
isa	"pine tree"
location	"sea mammal"
product	"protein production"
part-of	"human lung"

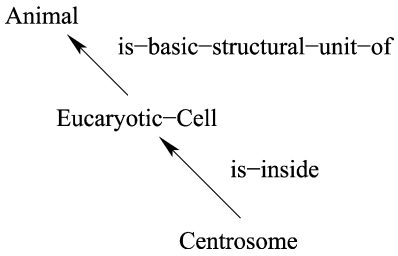


Fig. 11. The loose-speak interpretation of "Animal Centrosome".

6.2. Noun compounds

The computational linguistics community also has studied noun compound interpretation extensively [7,18,20,26,28,39, 41,42,62,65]. In these studies, noun compound interpretation is treated as a classification problem. The task is to select a single semantic category for each pair of nouns. The selection is usually made from a list of about 20 semantic categories, such as *part-of, material* and *object-of*. See Table 6 for examples.

Our task is more general. Rather than selecting a single semantic category, our task is to find a sequence of semantic relations that links the nouns in a compound. Semantic relations are a list of about fifty thematic roles such as *agent, object, has-part, location*, and so on [4]. For example, given *animal centrosome*, a traditional interpretation may classify this as a location category (an *animal centrosome* is a centrosome inside an animal). A loose-speak interpretation may be composed of a sequence of semantic relations, such as: "an *animal centrosome* is a centrosome that is inside a eucaryotic cell, which is the basic unit of an animal" (see Fig. 11).

Furthermore, computational linguists have approached the noun compound interpretation task armed with lots of examples, but little or no knowledge about the constituent nouns. Typical solutions are based on statistical patterns discovered in the corpus of examples [39]. In contrast, we approach the task in the context of deeper knowledge of the constituent nouns – their taxonomic classification, at least – but few examples of noun compounds, let alone a corpus.

6.3. Question answering

There has been a long history of research on question answering systems, and over the years, they have evolved into two main groups that address different types of questions.

- Knowledge-based question answering systems address questions that require reasoning and they return answers through inference. The loose-speak problem is very pertinent in these systems.
- Retrieval-based question answering systems answer factoid questions by retrieving relevant passages from a corpus. The misalignment problem between a question and a passage is a major challenge in these systems, and they adopted different approaches to address the issue.

6.3.1. Knowledge-based question answering

Nearly all of the earliest question answering systems [40,69,70] are knowledge based. They typically take in questions formulated in natural language and return concise answers. The systems typically consult knowledge bases which are manually built in symbolic logic, and thus support automated inference.

The knowledge bases used in these early systems were significantly smaller and simpler than the ones used today, therefore loose speak was less of a problem. For the few cases of loose speak, the early systems interpreted them as part of the natural language interpretation process using rules created manually – an approach that does not scale. In contrast, the Loose-Speak Interpreter automatically detects and interprets misalignment without using any manually crafted rules, and is shown to be effective on large knowledge bases.

6.3.2. Retrieval based question answering

As the amount of available on-line information increases, the scalability of automated question answering systems becomes more important. The Text Retrieval Conference [53] has greatly facilitated the development of large-scale question answering systems. These systems [63] answer factoid questions by combining information retrieval (IR) and shallow semantics to find precise answers in large corpora. They typically use information retrieval to find the relevant passages in the corpus, and they match the question with the retrieved passages.

Despite the small context provided by factoid questions and the redundancy in large corpora, misalignments between the question representation and the passage representation are one of the biggest challenges in these systems. These systems address the misalignment problem differently that we do because of different tolerances and resource availabilities. Retrieval-based systems are more tolerant of misalignments than are knowledge-based systems because they do not rely on logical inference. For retrieval-based systems, a misalignment may reduce the likelihood of the correct candidate being returned as the answer. However, for knowledge-based systems, the consequences are more serious: the logical reasoning process may fail completely if misalignments exist. In terms of resources, knowledge-based systems have deep domain knowledge, which retrieval-based systems lack.

For these reasons, retrieval-based systems address the misalignment problem differently than knowledge-based systems. The earlier retrieval based systems [51,58] relied on type matching (i.e. does the candidate passage provide an answer of the same type as expected by the question?) or light-weight inferencing (e.g. the density of terms that appear both in question and in passage). Later retrieval-based systems [30,50,67] employed syntactic and semantic features in a statistical translation or logic proving framework. In contrast, the Loose-Speak Interpreter exploits the domain knowledge to accurately detect and interpret misalignments between the question representation and the relevant knowledge base segments.

6.4. Expectation based knowledge acquisition

It has been long recognized that the knowledge in a knowledge-based system can be used to facilitate ongoing knowledge acquisition [17]. One way to take advantage of the existing knowledge is to derive expectations that guide knowledge acquisition [8,9,19,27,35,44,64]. Expectations are the predictions of what the user will enter next based on the user's past entries. Expectations have been shown to be effective in knowledge acquisition. In one study [35], the expectation-based knowledge acquisition tool provided a 30% savings for knowledge acquisition, and it detected, resolved or even avoided many mistakes early on.

Methods of expectation-based knowledge acquisition differ from loose-speak interpretation in two ways. First they differ in terms of detection. Expectation is used when a syntactic or a semantic error is detected. In contrast, occurrences of loose speak are entirely semantic. Although the Loose-Speak Interpreter only handles semantic issues, it uses a more active detection method. Loose speak is detected not only by a constraint violation, but also by the failure of the resemblance test (see Section 4.3). Our evaluation in Section 5.1.1.2 shows that constraint violations detect only 54% of the loose speak occurrences, and the rest are detected by the resemblance test. In addition to our empirical study, other researchers [46] have also shown that constraint violations, such as selectional restriction violations, are not sufficient to detect all occurrences of metonymy; other types of detection methods are needed as well.

Second, expectation-based knowledge acquisition methods and loose-speak interpretation differ in terms of search complexity. Because the expectation can be either syntactic based or semantic based, the consequent search is more free form. It is not as restricted by the triple representation as the ones used in the Loose-Speak Interpreter's test-and-repair routine. As a result, there may be more expectations found, and filtering out invalid expectations is an important task. The Loose-Speak Interpreter finds fewer interpretations, so ranking candidate interpretations is not as important.

7. Summary and future work

In this paper, we have addressed a key problem with knowledge-based systems: they are often brittle when given unanticipated input, i.e. assertions or queries that misalign with the ontology of the knowledge base. We have defined *naïve encoding* as an assertion or a query that is encoded without regard for the ontology being used. If, on the other hand, an encoding not only conveys the intended meaning of the input, but is also compatible with the knowledge base, then we call it a *correct encoding* of the input. An occurrence of *loose speak* is defined as a misalignment between a naïve encoding and its correct encoding.

Our thesis is that misalignments between naïve encodings and correct encodings occur frequently, and moreover with such regularity that the misalignments can be corrected algorithmically.

We conducted empirical studies in various domains and tasks to evaluate the thesis. First, we gauged the frequency of loose speak occurrences through two studies. We first analyzed naïve encodings of nearly 300 randomly chosen sentences from three corpora – the Brown corpus [37], Message Understanding Conference corpus [15,16] and a biology text book [1] – and we found that 80% of the sentence encodings contained at least one occurrence of loose speak. Second, we analyzed 146 naïve encodings of the Advanced Placement chemistry questions used for Project Halo, and we found that almost all of the question encodings contained at least one occurrence of loose speak. The results of these studies show that loose speak occurs frequently in knowledge base interactions.

Second, the regularity aspect of the thesis is addressed through the analysis of the misalignments found in the two studies. We found that they concentrated on a small number of types. There were six common types of loose speak, some of which occurred as frequently as 87% of the time.

Third, since the results of our analysis showed that loose speak occurs frequently and with regularity, we were able to develop an interpretation algorithm. Instead of developing separate algorithms to interpret each type of loose speak, we developed one algorithm to interpret all of the common types of loose speak.

We evaluated the Loose-Speak Interpreter with a variety of studies on different domains and tasks. The user study we conducted involved three users encoding a set of 50 Advanced Placement chemistry questions. We compared their naïve encodings with corresponding output from the Loose-Speak Interpreter. We found that 96% of the encodings on average contained loose speak, and the Loose-Speak Interpreter corrected these problems with about 95% precision and 90% recall. We applied the interpreter to 742 pairs of nouns in three domains: biology, engine repair, and Sparc Workstation. The interpreter performed with about 85% precision and 80% recall.

We also evaluated the effectiveness of the interpreter on encodings generated by a natural language processing system using the data from Project Halo. The data were generated by the natural language processing system applied to three domains (biology, chemistry and physics). The precision of the Loose-Speak Interpreter on these automatically generated encodings is similar to that on manually generated encodings, and the recall is lower mostly due to natural language processing errors. This suggests that the interpreter can handle human encodings as well as automatically generated correct encodings.

The results of these empirical studies show that we were able to exploit the regularity in loose speak occurrences to interpret them automatically, and that a wide variety of common types of loose speak could be handled with one parsimonious approach.

As for future work, there are additional extensions to the interpreter that can be explored. The current Loose-Speak Interpreter is implemented based on the assumption that the knowledge base being used is correct (although not necessarily complete). This assumption works well for carefully crafted knowledge bases, but it does not hold for knowledge bases that are automatically extracted from corpora. One extension of the interpreter is to detect and resolve loose speak using automatically extracted knowledge bases that contain incorrect or contradictory knowledge.

In our evaluations, human judges are used since there is no standard annotated benchmark data, and the correctness of an interpreter is relative to a specific knowledge base. In the future, creating such benchmark data sets would significantly facilitate the development and evaluation of other interpreters.

The Loose-Speak Interpreter can be used for a variety of applications. We list two possible areas here: ontology axiom mapping and knowledge-base interaction. The task of ontology axiom mapping involves expressing the axioms in one ontology (the source ontology) using the concepts and axioms in another ontology (the destination). The Loose-Speak Interpreter may be used for this task by viewing the source ontology's axioms as naïve encodings, and its interpretation is the mapped axioms.

Another application of the Loose-Speak Interpreter is knowledge-base interaction. Large knowledge-based systems that cover a variety of domains and reason on diverse topics are much needed, but they are not widely used. One major hurdle in using knowledge-based systems is the difficulty of interacting with them. This paper has provided a unified solution that enables an untrained user, either a human or a computer program, to successfully interact with a large knowledge base without knowing the underlying structure of the knowledge base.

Acknowledgements

We wish to thank Jason Chaw, Peter Clark, Yolanda Gil, Ken Murray, Dan Teccui, Peter Yeh and the anonymous reviewers for their insightful comments and suggestions.

References

[1] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Robert, P. Walter, K. Roberts, Essential Cell Biology: An Introduction to the Molecular Biology of the Cell, Garland Publishing Inc., 1998.
[2] J.D. Apresjan, I. Melchuk, A. Zholkovsky, Semantics and lexicography: Towards a new type of unilingual dictionary, in: Studies in Syntax and Semantics, 1969.
[3] H.F. Atkinson, Mechanics of Small Engines, McGraw–Hill, Inc., New York, 1990.
[4] K. Barker, Semi-automatic recognition of semantic relationships in English technical texts, PhD thesis, University of Ottawa, Ottawa, Ontario, 1998.
[5] K. Barker, A trainable bracketer for noun modifiers, in: Proceedings of the Twelfth Canadian Conference on Artificial Intelligence, Vancouver, 1998.
[6] K. Barker, B. Porter, P. Clark, A library of generic concepts for composing knowledge bases, in: Proceedings of First International Conference on Knowledge Capture, 2001.
[7] K. Barker, S. Szpakowicz, Semi-automatic recognition of noun modifier relationships, in: Proceedings of COLING-ACL '98, Montreal, 1998.
[8] W. Birmingham, G. Klinker, Knowledge acquisition tools with explicit problem-solving methods, The Knowledge Engineering Review 8 (1) (1993) 5–25.
[9] J. Blythe, Integrating expectations from different sources to help end users acquire procedural knowledge, in: Proceedings of Seventeenth International Joint Conference on Artificial Intelligence, 2001.
[10] T. Brown, H. LeMay, B. Bursten, J. Burdge, Chemistry: The Central Science, Pearson Education, Inc., Upper Saddle River, NJ, 2003.
[11] R. Browse, Knowledge identification and metaphor, in: Proceedings of the 2nd Biennial Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI-2), 1978, pp. 48–54.
[12] L. Chan, SPARCstation 1 Installation Guide, Sun Microsystems Inc., 1989.

[13] P. Clark, P. Harrison, T. Jenkins, J. Thompson, R. Wojcik, Acquiring and using world knowledge using a restricted subset of English, in: Proceedings of the 18th International FLAIRS Conference (FLAIRS'05), 2005.
[14] P. Clark, J. Thompson, B. Porter, Knowledge patterns, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference, 2000.
[15] DARPA, in: Proceedings of the 3rd Message Understanding Conference, 1991.
[16] DARPA, in: Proceedings of the 4th Message Understanding Conference, 1992.
[17] R. Davis, Interactive transfer of expertise: Acquisition of new inference rules, Artificial Intelligence 12 (2) (1979) 121–157.
[18] P. Downing, On the creation and use of English compounds, Language 53 (1977) 810–842.
[19] H. Eriksson, Y. Shahar, S.W. Tu, A.R. Puerta, M. Musen, Task modeling with reusable problem-solving methods, Artificial Intelligence 79 (1995) 293–326.
[20] C. Fabre, Interpretation of nominal compounds: Combining domain independent and domain-specific information, in: Proceedings of Sixteenth International Conference on Computational Linguistics, 1996, pp. 364–369.
[21] J. Fan, K. Barker, B. Porter, P. Clark, Representing roles and purpose, in: Proceedings of First International Conference on Knowledge Capture, 2001.
[22] J. Fan, B. Porter, Interpreting loosely encoded questions, in: Proceedings of Nineteenth National Conference on Artificial Intelligence, AAAI Press, 2004.
[23] R. Farkas, E. Simon, G. Szarvas, D. Varga, GYDER: Maxent metonymy resolution, in: Proceedings of SemEval 2007, 2007.
[24] D. Fass, Processing Metonymy and Metaphor, Ablex Publishing, Greenwich, Connecticut, 1997.
[25] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, The MIT Press, Boston, 1998.
[26] T. Finin, Constraining the Interpretation of Nominal Compounds in a Limited Context, Lawrence Erlbaum Associates, NJ, 1986.
[27] Y. Gil, E. Melz, Explicit representations of problem-solving strategies to support knowledge acquisition, in: Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996.
[28] R. Girju, D. Moldovan, M. Tatu, On the semantics of noun compounds, Computer Speech and Language 19 (4) (2005) 479–496.
[29] B. Grosz, D. Appelt, P. Martin, F. Pereira, TEAM: An experiment in the design of transportable natural-language interfaces, Artificial Intelligence 32 (2) (1987) 173–243.
[30] C. Hang, R. Sun, K. Li, M.Y. Kan, T.S. Chua, Question answering passage retrieval using dependency relations, in: Proceedings of 28th Annual International ACM SIGIR 2005, 2005.
[31] S. Hartrumpf, J. Leveling, Combining approaches for classifying metonymy of named locations, in: J. Neves, M. Santos, J. Machado (Eds.), New Trends in Artificial Intelligence, Benjamins, 2007, pp. 767–778.
[32] J. Hobbs, Granularity, in: Proceedings of the Ninth International Joint Conference on Artificial Intelligence 1985, 1985.
[33] E. Iverson, S. Helmreich, Metallel: An integrated approach to non-literal phrase interpretation, Computational Intelligence 8 (3) (1992) 477–493.
[34] D. Jurafsky, J. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, Prentice Hall, NJ, 2000.
[35] J. Kim, Y. Gil, Deriving expectations to guide knowledge-base creation, in: Proceedings of Sixteenth National Conference on Artificial Intelligence, AAAI Press, 1999.
[36] P. Koch, Frame and contiguity: On the cognitive basis of metonymy and certain types of word-formation, in: K.U. Panther, G. Radden (Eds.), Metonymy in Language and Thought, Benjamins, 1999, pp. 139–167.
[37] H. Kucera, W.N. Francis, Brown corpus manual, Technical report, Brown University, Department of Linguistics, 1979.
[38] G. Lakoff, M. Johnson, Metaphors We Live By, University of Chicago Press, Chicago, 1980.
[39] M. Lauer, M. Dras, A probabilistic model of compound nouns, in: Proceedings of the 7th Australian Joint Conference on Artificial Intelligence, World Scientific Press, 1994.
[40] W. Lehnert, The Process of Question Answering, PhD thesis, Yale University, New Haven, CT, 1977.
[41] R. Leonard, The Interpretation of English Noun Sequences on the Computer, Elsevier Science, Amsterdam, 1984.
[42] J. Levi, The Syntax and Semantics of Complex Nominals, Academic Press, New York, 1979.
[43] S. Lytinen, R. Burridge, J. Kirtner, The role of literal meaning in the comprehension of non-literal constructions, Computational Intelligence 8 (3) (1992) 416–432.
[44] S. Marcus, J. McDermott, SALT: A knowledge acquisition language for propose-and-revise systems, Artificial Intelligence 39 (1) (1989) 1–37.
[45] K. Markert, U. Hahn, On the interaction of metonymies and anaphora, in: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, 1997.
[46] K. Markert, U. Hahn, Understanding metonymies in discourse, Artificial Intelligence 135 (2002) 145–198.
[47] K. Markert, M. Nissim, Metonymy resolution as a classification task, in: D. Yarowsky (Ed.), Proceedings of EMNLP2002, 2002.
[48] K. Markert, M. Nissim, Towards a corpus annotated for metonymies: The case of location names, in: Proceedings of LREC2002, 2002, pp. 1385–1392.
[49] K. Markert, M. Nissim, Semeval-2007 task 08: Metonymy resolution at SemEval-2007, in: Proceedings of SemEval 2007, 2007.
[50] D. Moldovan, C. Clark, S. Harabagiu, COGEX: A logic prover for question answering, in: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, 2003.
[51] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, V. Rus, LASSO: A tool for surfing the answer net, in: Proceedings of Text REtrieval Conference (TREC-8), 1999.
[52] C. Nicolae, G. Nicolae, S. Harabagiu, UTD-HLT-CG: Semantic architecture for metonymy resolution and classification of nominal relations, in: Proceedings of SemEval 2007, 2007.
[53] A. NIST, Text retrieval conference, http://trec.nist.gov, 2005.
[54] G. Nunberg, The Pragmatics of Reference, Indiana University Linguistics Club, IN, 1978.
[55] A. Ortony, Some psycholinguistic aspects of metaphor, in: R. Honeck, R. Hoffman (Eds.), Cognition and Figurative Language, Erlbaum Associates, 1980, pp. 69–83.
[56] Y. Peirsman, D. Geeraerts, Metonymy as a prototypical category, Cognitive Linguistics 17 (3) (2006) 269–316.
[57] B. Porter, P. Clark, KM—The knowledge machine: User manual, Technical report, University of Texas at Austin, http://www.cs.utexas.edu/users/mfkb/km.html, 1998.
[58] J. Prager, E. Brown, A. Coden, D. Radev, Question-answering by predictive annotation, in: Proceedings of the 23rd Annual International ACM SIGIR 2000, 2000.
[59] R. Schrag, M. Pool, V. Chaudhri, R. Kahlert, J. Powers, P. Cohen, J. Fitzgerald, S. Mishra, Experimental evaluation of subject matter expert-oriented knowledge base authoring tools, 2001.
[60] H.L. Somers, Valency and Case in Computational Linguistics, Edinburgh University Press, 22 George Square, Edinburgh, 1987.
[61] J. Sowa, Conceptual Structures: Information Processing in Mind and Machine, Addison–Wesley Publishing Company, New York, 1984.
[62] W. Ter Stal, Automated semantic analysis of compounds: Problem identification and literature overview, PhD thesis, University of Twente, The Netherlands, 1996.
[63] T. Strzalkowski, S. Harabagiu, Advances in Open Domain Question Answering, Springer, 2003.
[64] W. Swartout, Y. Gil, EXPECT: Explicit representations for flexible acquisition, in: Proceedings of the Ninth Knowledge-Acquisition for Knowledge-Based Systems Workshop, 1995.

[65] L. Vanderwende, Algorithm for automatic interpretation of noun sequences, in: Proceedings of Fifteenth International Conference on Computational Linguistics, 1994, pp. 782–788.
[66] Vulcan Inc., Project Halo, http://projecthalo.com, 2003.
[67] M. Wang, N.A. Smith, T. Mitamura, What is the Jeopardy model? A quasi-synchronous grammar for QA, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007, pp. 22–32.
[68] R. Weischedel, N. Sondheimer, Meta-rules as a basis for processing ill-formed input, American Journal of Computational Linguistics 9 (3–4) (1983) 161–177.
[69] T. Winograd, Understanding Natural Language, Academic Press, New York, 1972.
[70] W. Woods, R. Kaplan, B. Webber, The lunar sciences natural language information system: Final report, Technical Report BBN Report 2378, Bolt Beranek and Newman Inc., Cambridge, MA, 1972.